黒点ランダム抽出と Hough 曲面の交点計算による円図形検出の一手法†

塩野 充††

ハフ変換は画像中に存在する直線を安定に検出する有力な特徴抽出手法として考案された．ハフ変換を用いれば，局所的方法では検出困難な低品質画像等における断続の多い不安定な直線をも安定に検出することができる．さらに，直線だけではなく，円や楕円，その他の一般的な図形を抽出しようとするハフ変換の研究も行われている．画像からの直線，円，楕円，その他の図形の抽出はパターン認識の基本的な問題であり，画像理解の分野で極めて重要な処理要素となりつつある．ハフ変換の基本的な問題点としては，第1は処理速度の問題であり，種々の高速化手法が提案されている．第2はパラメータ空間に必要とするメモリの問題である．検出図形の精度を高めるにはパラメータを細かく計算する必要があり，パラメータ空間を大きく設定しなければならず，そのための大きなメモリが必要となる．また，検出図形が複雑化すると，パラメータ空間の次元数が高くなり，必要メモリが一挙に増大する．本論文では，画面中の黒点を乱数によって順次抽出し，各黒点に対応するハフ曲面の全交点を代数的に求め，その座標値を指定桁で丸めてから併合するという処理を繰り返すことにより，パラメータ空間を使用せずに少ないメモリで任意の高精度で図形の検出が可能な方法を提案し，途切れや重なり，雑音のある複数個の円の検出実験を行った．

1. はじめに

Hough（ハフ）変換はディジタル画像中に存在する直線等を安定に検出する有力な特徴抽出手法として考案され[1),2)]，近年その応用が活発化している[3),4)]．Hough 変換を用いれば，細線化追跡法等の局所的方法では検出困難な低品質画像等における断続（途切れ）の多い不安定な直線成分をも安定に検出することができる．さらに，直線だけではなく，円や楕円，その他の一般的な図形を抽出しようとする Hough 変換の研究も行われている[5)~12)]．ディジタル画像からの直線，円，楕円，その他の図形の抽出はパターン認識の基本的な問題であるが，近年，コンピュータビジョン，ロボットビジョン等の3次元的な画像理解の分野で極めて重要な処理要素となりつつある．本論文は Hough 変換を用いて，円や楕円等を効率よく検出することを目標としている．

Hough 変換の基本的な問題点としては2つが考えられる．第1は計算量の多さに起因する処理速度の問題であり，種々の高速化手法が提案されている[13)~16)]．第2はパラメータ空間に必要とするメモリの問題である．検出しようとする図形の精度を高めるにはパラメータを細かく計算する必要があり，パラメータ空間を非常に大きく設定しなければならず，そのための大きな配列用のメモリが必要となる．また，検出しようとする図形が複雑化すると，図形を表すパラメータが多くなり，パラメータ空間の次元数が高くなり，必要メモリが一挙に増大する．本論文では，画面中の黒点を乱数によって順次抽出し，各黒点に対応するハフ曲面の全交点を代数的に求め，その座標値を指定桁で丸めてから併合するという処理を繰り返すことにより，パラメータ空間を使用せずに少ないメモリで任意の高精度で図形の検出が可能な方法[17)]を示し，途切れや重なり，雑音のある複数個の円の検出実験を行った．

2. Hough 変換による円の検出

通常の Hough 変換による円の検出方法を以下に示す．xy 平面上にある，中心が (a,b) で，半径が r (>0) の円の方程式は，

$$(x-a)^2+(y-b)^2=r^2 \qquad (1)$$

と表される．

画面サイズ $X\times Y$ の2値原画像を $F(x,y)$ $(x=0\sim X-1;\ y=0\sim Y-1)$ とし，その中の N 個の黒点（エッジ点）を $P_n(x_n,y_n)$ $(n=1\sim N)$ とする．それらの黒点が式(1)の円を構成しているならば，

$$(x_n-a)^2+(y_n-b)^2=r^2 \qquad (2)$$

となる．ここで逆に，x_n，y_n を定数と考え，a，b，r を変数と考えると，式(2)は (a,b,r) の3次元空間（これをパラメータ空間と呼ぶ）の中にある曲面（Hough 曲面と呼ぶ）を表している．この曲面は具体

† A Circle Detection Method Based on the Random Calculation of the Intersections among Hough Surfaces by MITSURU SHIONO (Department of Electronic Engineering, Faculty of Engineering, Okayama University of Science).

†† 岡山理科大学工学部電子工学科

的には点 $(x_n, y_n, 0)$ を頂点とし，中心軸が r 軸に平行で，r (≥ 0) に比例して直径が大きくなってゆく円錐面を表している．1個の黒点 P_n に関する Hough 曲面を3次元配列 $G(\alpha, \beta, \gamma)$ $(\alpha=0\sim\alpha_{MAX}; \beta=0\sim\beta_{MAX}; \gamma=0\sim\gamma_{MAX})$ で表されたパラメータ空間に写像する．ここで，α は a 軸に，β は b 軸に，γ は r 軸に対応し，α_{MAX} は a 軸の刻み数，β_{MAX} は b 軸の刻み数，γ_{MAX} は r 軸の刻み数を表す．

処理は，a を副走査で0から逐次，$\Delta a, 2\Delta a, 3\Delta a, \cdots, a_{MAX}(=\alpha_{MAX}\Delta a)$，$b$ を主走査で0から逐次，$\Delta b, 2\Delta b, 3\Delta b, \cdots, b_{MAX}(=\beta_{MAX}\Delta b)$ と増加させていって，その各々の場合の r の値を式(2)で計算して，配列 G 上の対応する $G(\alpha, \beta, \gamma)$ の要素値を1だけ増加させる．求めた r も，$\Delta r, 2\Delta r, 3\Delta r, \cdots, r_{MAX}(=\gamma_{MAX}\Delta r)$ なる刻み間隔のいずれかに量子化するものとする．なお，配列 G の初期値はすべて0とする．

すべての黒点に対してこの計算を行うと，配列 G の3次元空間内にはすべての Hough 曲面が要素値をもった voxel の集合で描かれ，複数の Hough 曲面が交わっている部分では G の要素値が，交点の度数分布を表していることになる．これより，度数分布の最も大きい交点の (a, b, r) を求めると，式(1)から検出すべき円の方程式が得られる．

上記の方法は Hough 変換による円抽出の最も基本的な方法であるが，上述したように a，b，r の検出精度を上げようとして，a，b，r 軸の刻み幅 Δa, Δb, Δr を小さくすると，パラメータ空間を表す配列 G の要素数が一挙に増大し，非常に大きなメモリが必要となってしまう．

本研究では，a，b，r の検出精度を上げるために，a，b，r の刻み幅をいくら小さくしても必要メモリ量が増大しない方法として，パラメータ空間を使わずに，少ないメモリで高精度な計算が可能な方法を提案し，基礎的な実験を行った．次章以下にその説明を行う．

3. 黒点ランダム抽出と丸め併合

円を抽出するための通常の Hough 変換計算法は円を特定する3つのパラメータ a，b，r のうちの2つのパラメータ a，b に関してすべてのパラメータ空間を2次元走査して，その各々の位置における r の値を求めて3次元座標 (a, b, r) の要素値（度数）を1だけ増加させてゆき，最終的に最も大きな度数の点を求める方法である．しかし，本方法ではパラメータ空間を使用せずに，任意の3つの Hough 曲面の交点を代数的に求めることを基本にしている．すべての Hough 曲面の3つの組合せに関して，その交点位置をすべて求め，その座標をクラスタリングすることにより，最大クラスタのサンプル数を求める．このサンプル数が度数に等価なものとなる．しかし，ここで実際に通常のクラスタリング計算を行えばまた，膨大な3次元の距離行列を必要とし，計算時間も非常に長くなるのでメリットがなくなる．本方式においては，座標位置を適当な桁数で丸め（四捨五入）を行うことにより，a，b，r 共に等しくなった2つの交点座標を順次併合していく簡単な方法を用いている．これにより，クラスタリングを行うことなく，近傍の交点をまとめてゆくことができる．本方式で必要とする配列（メモリ）は後述するように黒点数の3乗のオーダで増加する．したがって，黒点数が多いと必要メモリも大きくなり，パラメータ空間を用いないという長所が薄れてしまう．そこで，本方式では一度に画面上のすべての黒点を用いて計算を行うのではなく，乱数によって適当数の黒点をサンプリングし，交点計算と丸め併合処理を行う．この過程を原画面上の黒点がなくなるまで繰り返し処理を行い，そのすべてが終った段階で度数の合計を求めて，度数の大きいものを上位から所望の数だけ採用して最終結果を求める方式（黒点ランダム抽出法）を採用している．乱数による黒点のサンプリングについては第4章で説明する．

3.1 処理の手順

図1に本方式（黒点ランダム抽出法）の処理のジェネラルフローを示す．

(a)では処理対象となる2値画像を配列 F に入力する．

(b)では画面 F を走査し，対象とする黒点（エッジ点）を乱数で抽出して，その xy 座標値を配列 P に記録する．P は黒点位置表と呼ぶ．一度記録した黒点は画面 F から消去する（-1 にする）．

(c)では任意の3つの黒点に対して，それに対応する3枚の Hough 曲面の交点座標 (a, b, r) を求めて，それを配列 C に記録する．すべての黒点の3個の組合せに対して計算を行う．配列 C は交点位置表と呼ぶ．

(d)では交点位置表 C に格納された a，b，r に関して，必要な精度で指定された桁で丸め（四捨五入）処理を行う．

(e)では丸め処理した結果，a，b，r とも等しくなった交点を順次併合してゆく．この時，併合回数も

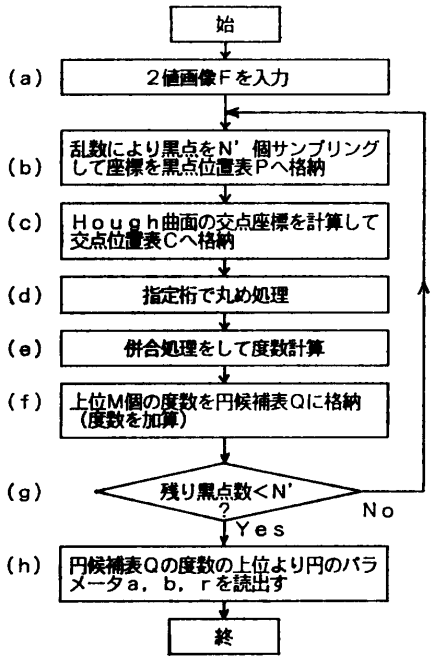

図 1 本方式のジェネラルフロー
Fig. 1 General flowchart of the proposed method.

記録してゆく．これが度数 h となる．

(f)では併合回数，すなわち度数 h の大きなものを上位 M 個（M を抽出したい円の数の数倍とする．実験では $M=30$）抽出して，その a，b，r，h を表 Q（$M\times 4$ の2次元配列）に記録する．表 Q は円候補表と呼ぶ．

(g)では画面 F 上の残り黒点数が N' より少なくなるまで(b)～(f)を繰り返させる．

(h)では円候補表 Q より，度数 h の大きい順に第1位から抽出したい円の数だけパラメータ a，b，r を読み出す．これが最終結果となる．

3.2 交点の計算

今，具体的に3つの黒点 $P_i(x_i, y_i)$，$P_j(x_j, y_j)$，$P_k(x_k, y_k)$ が式(2)の円上にあるとすると，

$$\left.\begin{aligned}(x_i-a)^2+(y_i-b)^2&=r^2\\(x_j-a)^2+(y_j-b)^2&=r^2\\(x_k-a)^2+(y_k-b)^2&=r^2\end{aligned}\right\} \quad (3)$$

となる．これを a，b，r について解くと，

$$a=\{X_i^{(2)}Y_j+Y_i^{(2)}Y_j-X_j^{(2)}Y_i-Y_j^{(2)}Y_i\}/\{2(X_iY_j-X_jY_i)\} \quad (4)$$

$$b=\{X_j^{(2)}+Y_j^{(2)}-2aX_j\}/(2Y_j) \quad (5)$$

$$r=\{(x_i-a)^2+(y_i-b)^2\}^{1/2} \quad (6)$$

ただし，

$$\left.\begin{aligned}X_i&=x_i-x_j\ ,\ Y_i=y_i-y_j\\X_j&=x_j-x_k\ ,\ Y_j=y_j-y_k\\X_i^{(2)}&=x_i^2-x_j^2,\ Y_i^{(2)}=y_i^2-y_j^2\\X_j^{(2)}&=x_j^2-x_k^2,\ Y_j^{(2)}=y_j^2-y_k^2\end{aligned}\right\} \quad (7)$$

となる．式(5)，(6)の b，r は式(4)の a を解けば得られる．式(4)～(6)より得られた解 (a, b, r) は式(3)で表現される3つの円錐面の交点座標を表している．

3.3 交点位置表の作成

N 個の Hough 曲面の交点数 N_H は3個ずつの組合せの総数ゆえ，

$$N_H={}_NC_3=N(N-1)(N-2)/6 \quad (8)$$

となる．交点位置格納用の2次元配列（交点位置表と呼ぶ），

$$C(1:4,\ 1:N_H)$$

を用意し，式(4)～(6)によって求めた N_H 個の交点位置の a を $C(1,*)$ に，b を $C(2,*)$ に，r を $C(3,*)$ に格納する．ここで，$*$ はすべての添字を表す．すなわち配列の断面を表す．$C(4,*)$ は後述するように度数を格納するための場所である．複数個の黒点によって構成される Hough 曲線の交点を求める方法は興水の HCGA アルゴリズム[13]において，直線の検出のために示されているが，本計算方法は円検出のためのその Hough 曲面への適用とも考えられる．

3.4 丸めと併合

すべての $C(1,*)$，$C(2,*)$，$C(3,*)$ を有効桁が，それぞれ小数第 μ 位，小数第 λ 位，小数第 ν 位になるように四捨五入する．すなわち，a は小数第 $\mu+1$ 位で，b は小数第 $\lambda+1$ 位で，r は小数第 $\nu+1$ 位で四捨五入する．μ，λ，ν は必要とする a，b，r の精度によって決める四捨五入の桁数で，精度が細かくなるほど大きくなる．例えば，$C(1,n)$ が 1.234567 のとき，$\mu=4$ で四捨五入を行うと，$C(1,n)=1.2346$ となる．

この後，交点の度数を計算して $C(4,*)$ へ格納する．すなわち，ある2つの交点の a，b，r がともに一致すれば，その2つの交点は併合可能と見なし，$C(4,*)$ の値を1だけ増加する．なお，$C(4,*)$ の初期値はすべて1としておく．度数計算のアルゴリズムを以下に示す．∧は and を表し，←は代入を表す．また，−999 は処理済みを表すフラグの意味とする．$i=1\sim N_H-1$，$j=i+1\sim N_H$ の2重の DO ループとする．

$$\left.\begin{array}{l}\text{if}\quad C(1,i)\neq -999\\ \quad\wedge\ C(1,i)=C(1,j)\\ \quad\wedge\ C(2,i)=C(2,j)\\ \quad\wedge\ C(3,i)=C(3,j)\\ \text{then}\ C(4,i)\leftarrow C(4,i)+1\\ \qquad C(1,j)\leftarrow -999\end{array}\right\}\qquad (9)$$

3.5 円の検出

交点位置表 C に丸め併合処理を行った結果，度数 $C(4,*)$ より円を検出できる．円の検出には2通りの方法がある．1つは検出する円の個数が決まっている場合で，度数の最大のものから順にその個数だけ抽出すればよい．もう1つは度数のしきい値を $\tau\ (>1)$ として，τ 以上の度数の円を検出する方法である．この場合は検出される円の個数は τ によって変化する．いずれの場合も，検出される円の方程式は次のようになる．

$$\left.\begin{array}{l}(x-a_m)^2+(y-b_m)^2=r_m^2\\ \text{ただし，}\ a_m=C(1,m)\\ \qquad b_m=C(2,m)\\ \qquad r_m=C(3,m)\\ \qquad 1\leqq m\leqq N_H\end{array}\right\}\qquad (10)$$

4. 乱数による黒点のサンプリング

上述の 3.2〜3.4 節の計算方式で必要とする主なメモリを配列要素の数で計算すると，黒点数を N 個とした場合，黒点位置表 P のためには，$2N$ 要素，交点位置表 C のために $4N_H=4\cdot{}_NC_3=4\cdot N(N-1)(N-2)/6$ 要素必要になるので，1要素を4バイトとすると合計で，

$$8(N^3-3N^2+5N)/3\quad(\text{バイト})\qquad (11)$$

となり，大体 N の3乗のオーダで増加する．一方，パラメータ空間を用いる通常の方法では，パラメータ空間に要するメモリは，$G\,(0:\alpha_{MAX},\ 0:\beta_{MAX},\ 0:\gamma_{MAX})$ なる3次元配列ゆえ，ほぼ，$4\alpha_{MAX}\cdot\beta_{MAX}\cdot\gamma_{MAX}$ (バイト) となる．したがって上述の方式がメモリ節減の面で有効になるには，

$$8(N^3-3N^2+5N)/3\ll 4\alpha_{MAX}\cdot\beta_{MAX}\cdot\gamma_{MAX}\qquad (12)$$

とならなければならない．いま仮に，α_{MAX}，β_{MAX}，γ_{MAX} を画面 F の一辺 $X\,(=Y)$ の大きさに合わせるとすると，式(12)は，

$$N^3-3N^2+5N\ll(3/2)X^3\qquad (13)$$

この式から大体分かるように黒点数 N は画面サイズ X とコンパラブルではだめで，N は X よりはるかに小さくなければならない．しかし，一般には画面上の黒点数はそのように少ないとは言えず，ほとんどの場合，式(13)は成り立たないと考えるのが自然であろう．

そこで，本方式においては，原画面 F 上にあるすべての黒点を一度に使うのではなく，計算時における黒点数が小さくなるように何回にも分けて計算を行う方法を採用している．すなわち，黒点をサンプリングして使用する．一度サンプリングして使った黒点は以降の回では使わないようにして，最終的にはほぼすべての黒点をまんべんなく使用するようにしている．

1回にサンプリングする黒点数を $N'\ (<N)$ とする．サンプリングは，走査（画面左上から横走査とする）する順に出現する黒点を N' 個取り出したのでは画面 F のほんの一部分の状況しか反映していないことになる．そこで，1回のサンプリングに画面全体の性質（黒点の分布状況）が反映するように，乱数を用いて次の①〜③の手順に従ってサンプリングを行う（黒点ランダム抽出法）．これにより1回の走査は必ず画面全体を被覆することになる．R_n は区間 $[0,1]$ の一様乱数とする．P は黒点位置表である．

① $n\leftarrow 0$

② 次の処理を x，y に関する走査にて行う．

$$\left.\begin{array}{l}\text{if}\ F(x,y)=1\wedge R_n\leqq\Phi\\ \text{then}\ n\leftarrow n+1\\ \qquad P(1,n)\leftarrow x,\quad P(2,n)\leftarrow y\\ \qquad F(x,y)\leftarrow -1\end{array}\right\}\qquad (14)$$

③ 走査途中で $n\geqq N'$ になれば終了し，走査を終えてもまだ $n<N'$ ならば，もう一度②へ行く．

ここで，F の画素値の -1 はサンプリング済みの黒点を表すフラグとする．Φ は乱数しきい値で，その黒点を抽出するかしないかを決めるキーとなる．すなわち，1回にサンプリングする黒点数を N'，その回のサンプリングの走査を開始する時点でまだ画面に残っている黒点数を B とすると，次式で定義される値である．

$$\Phi=N'/B\qquad (15)$$

この乱数しきい値の設定により，1回のサンプリングはほぼ画面全体を走査し終えた時点で終了する．したがって，画面全体の性質（黒点の分布状況）が反映される（もちろん，図形的にも反映されるという厳密な証明はできないが，実用的には実験結果から見て，サンプリング黒点数 N' が少なすぎない限り，反映されると考えてよいと思われる）．

最終回のサンプリングにおいて残りの黒点数 B が，$B<N'$ となった場合にはその少ない黒点数で無理にもう1行程を行うと不正確な結果が出て，かえって全体に悪影響を及ぼすので，それらは切り捨てて終了す

る．

5. 識別実験

5.1 原画像の作成

画面サイズ $X\times Y$ の原画像を $F(0:X-1,\ 0:Y-1)$ とし，その中に次に示す大小5個の円を描き，それらを検出する実験を行った．(a,b) は円の中心で，r は半径である．

$$\left.\begin{array}{l}\text{円 \#1}: a=60,\ b=60,\ r=50\\ \text{円 \#2}: a=80,\ b=120,\ r=70\\ \text{円 \#3}: a=140,\ b=70,\ r=40\\ \text{円 \#4}: a=200,\ b=50,\ r=20\\ \text{円 \#5}: a=150,\ b=150,\ r=100\end{array}\right\} \quad (16)$$

これらの円の描画は式(17)のような手順を用いた．ここで，$x=0\sim X-1$; $y=0\sim Y-1$ とする．$\leftarrow$ は代入を表す．F の初期値はすべて0とする．

$$\text{if } |(x-a)^2+(y-b)^2-r^2|\leqq\varepsilon^2$$
$$\text{then } F(x,y)\leftarrow 1 \quad (17)$$

具体的には $X=Y=256$, $\varepsilon=5$ とした．総黒点数は851個となった．しきい値 ε の設定により，円環の太さが制御されるが，ここではカスレも生じるようにして円の一部に隠蔽のある状況も表現するようにした．

5.2 実験結果と検討

実験は東京大学大型計算機センターのスーパコンピュータ S820/80 および大型汎用機 HITAC/M 682 を使用した．スーパコンピュータを使用したのは計算コストの節約だけの理由で，スーパコンピュータを使用しなければできないというわけではない．プログラムは FORTRAN で記述した．実験における画像 F の画面サイズ $X\times Y$ は，256×256 とした．これと併せて，パラメータ空間を使用する通常の Hough 変換計算法による実験も行った．

本方式の実験においては，1回のサンプリングに使用する黒点数 N' は，8, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100 の 11 通りで実験を行った．各々の場合の実験結果を**表1**に示す．総黒点数 $N=851$ 個である．画面中に乱数によって雑音を付加した場合の実験も行った．雑音の黒点数 Z は，雑音比率 Z/N が 0.0 ($Z=0$), 0.5 ($Z=425$), 1.0 ($Z=851$) の3つの場合について行った．すなわち，雑音が全くない場合，雑音が総黒点数の半分の場合，雑音が総黒点数と同じだけある場合である．**図2**(a), (b), (c)にこれらの3種類の画像を示す．サンプリングの総ループ回数は $(N+Z)/N'$ の小数以下を切り捨てた値となる．なお，前述の四捨五入の桁数 μ, λ, ν は a, b, r の検出精度を原画像 F 上の1画素単位（整数値）に設定したことから，μ, λ, ν ともに0とした．

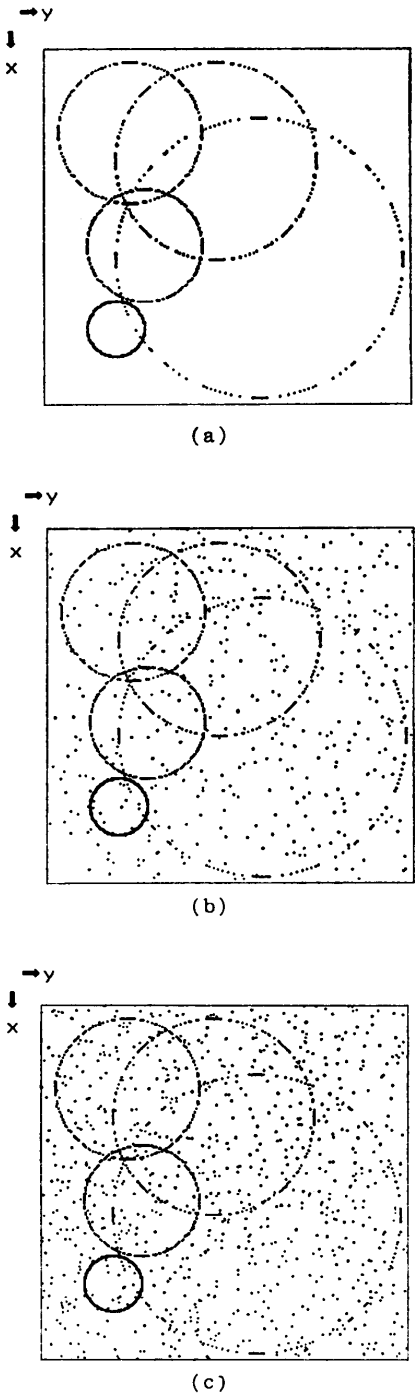

図2 実験に用いた原画像 ((a) $Z/N=0.0$, (b) $Z/N=0.5$, (c) $Z/N=1.0$.)

Fig. 2 The original image used in the experiment. ((a) $Z/N=0.0$, (b) $Z/N=0.5$, (c) $Z/N=1.0$.)

表 1 本方式による円の検出実験結果 ($X=Y=256$, $N=851$)
Table 1 The result of the circles detection experiment by the proposed method ($X=Y=256$, $N=851$).

サンプリング黒点数	雑音対総黒点数比率	円検出結果 (上段は円番号, ＊は偽円, 下段は度数の%)						CPU 時間
N'	Z/N	第 1 位	第 2 位	第 3 位	第 4 位	第 5 位	第 6 位	(秒)
8	0.0	#5 13.684	#3 11.842	#4 11.184	#1 10.526	#2 8.026	＊ 3.026	1.921
	0.5	#5 12.079	#2 7.327	#3 4.554	＊ 4.158	#1 3.960	＊ 3.564	2.866
	1.0	#5 6.726	#2 5.841	#＊ 5.310	#1 5.310	＊ 4.779	＊ 4.248	3.857
10	0.0	#5 13.451	#3 11.781	#2 10.019	#1 9.833	#4 9.184	＊ 3.061	1.444
	0.5	#5 12.620	#2 9.615	#3 9.615	#1 4.928	#4 4.207	＊ 2.885	2.199
	1.0	#5 7.503	#1 6.027	#2 5.166	＊ 4.428	＊ 4.182	#3 3.813	2.998
20	0.0	#2 16.683	#5 14.000	#3 13.765	#4 10.417	#1 8.576	＊ 2.643	1.540
	0.5	#5 14.532	#3 9.032	#2 8.951	#1 8.385	#4 4.476	＊ 3.586	2.291
	1.0	#5 11.338	#2 7.223	#1 7.132	#3 6.614	＊ 3.810	＊ 3.749	3.153
30	0.0	#5 16.773	#3 15.741	#2 12.237	#4 9.631	#1 9.365	＊ 2.447	2.839
	0.5	#5 14.470	#2 10.475	#1 9.558	#3 7.212	#4 4.541	＊ 3.600	4.403
	1.0	#5 14.568	#1 8.697	#2 8.272	#3 4.019	＊ 3.745	＊ 3.114	5.943
40	0.0	#5 17.195	#3 15.321	#2 13.995	#1 8.534	#4 6.410	＊ 2.870	6.032
	0.5	#5 15.395	#2 11.040	#3 10.883	#1 7.848	#4 6.778	＊ 3.954	9.389
	1.0	#5 13.851	#2 7.980	#1 7.558	#3 7.422	＊ 3.368	＊ 3.296	12.972
50	0.0	#5 16.393	#3 15.345	#2 13.120	#1 9.501	#4 8.613	＊ 3.396	12.256
	0.5	#5 14.136	#2 11.750	#1 10.508	#3 9.408	#4 5.173	＊ 4.074	18.784
	1.0	#5 14.282	#2 8.759	#1 8.676	#3 6.408	＊ 3.380	＊ 3.195	25.932
60	0.0	#5 16.200	#3 15.479	#2 15.135	#4 8.838	#1 8.719	＊ 2.464	21.344
	0.5	#5 13.624	#2 13.196	#3 9.408	#1 8.598	#4 6.087	＊ 3.294	34.172
	1.0	#5 15.321	#2 8.001	#1 5.719	#3 5.576	＊ 3.596	＊ 3.405	47.206
70	0.0	#5 15.572	#3 15.426	#2 14.729	#1 10.343	#4 7.915	＊ 2.807	36.242
	0.5	#5 14.611	#2 13.882	#3 9.292	#1 8.571	#4 4.515	＊ 3.358	58.453
	1.0	#5 14.196	#2 10.370	#1 7.457	#3 6.301	＊ 4.132	＊ 3.649	79.839
80	0.0	#5 14.903	#3 14.766	#2 13.638	#1 13.350	#4 7.407	＊ 2.944	56.226
	0.5	#5 14.450	#2 14.281	#1 10.288	#3 9.479	#4 5.287	＊ 2.848	89.677
	1.0	#5 13.964	#2 8.288	#1 6.735	#3 5.896	＊ 4.115	＊ 4.106	130.215

表 1（続き）

サンプリング黒点数 N'	雑音対総黒点数比率 Z/N	円検出結果（上段は円番号，＊は偽円，下段は度数の%） 第1位	第2位	第3位	第4位	第5位	第6位	CPU 時間（秒）
90	0.0	#5 15.744	#2 14.332	#3 13.444	#1 12.347	#4 8.099	＊ 2.944	91.245
	0.5	#5 15.748	#2 10.894	#1 10.750	#3 9.378	#4 4.726	＊ 4.053	152.998
	1.0	#5 14.706	#1 7.730	#2 7.582	#3 5.493	＊ 3.817	＊ 3.732	203.378
100	0.0	#5 15.686	#2 14.750	#3 14.013	#4 8.751	#1 8.625	＊ 3.057	129.783
	0.5	#5 14.035	#2 12.003	#3 9.138	#1 8.839	#4 4.849	＊ 3.817	212.757
	1.0	#5 15.219	#2 9.133	#1 7.433	#3 6.668	＊ 4.387	＊ 3.819	314.176

円の検出結果の欄の上段は式(16)の円番号であり，下段は度数の大きい順に第1位から第 30 位までを並べた場合の，度数合計のうちに占める割合をパーセントで示したものである．

雑音のない（$Z/N=0.0$）ときには，$N'=8$～100（N'/N では約 1～12%）のいずれの場合も第1位から第5位までに式(16)で設定した5つの円が正しく検出されていることが分かる．なお，$N'=8$ は正しく5個の円が検出される限度であり，これ以上少なくすると誤りが発生した．また，検出された円の第6位以下は雑音としての円（偽円）であり，度数の低い方が望ましいが，いずれの場合も第1位から第5位までに比べると，かなり低い値となっており，本方式の偽円抑圧性能が良好なことを示している．処理に要する CPU 時間は S820/80 を使用した場合で，N' が 10 個ずつ大きくなるにつれ，数倍に大きくなってゆくので円検出性能に支障のない限り，なるべく小さな N' を用いる方がよい．

雑音が黒点数の半分（$Z/N=0.5$）のときには，$N'=8$ の場合以外のすべての場合で，第1位から第5位までに式(16)で設定した5つの円が正しく検出されていることが分かる．

雑音が黒点数と同じ（$Z/N=1.0$）ときには，$N'=8$，10 の場合以外のすべての場合で，第1位から第4位までに式(16)で設定したうちの #4 以外の4つの円が正しく検出されていることが分かる．したがって，雑音黒点が多い場合には N' はなるべく大きい方が信頼性が向上することは明らかである．

表 2 にパラメータ空間を用いる通常の Hough 変換の計算方法で同じ画像 F で円の検出を行った場合の結果を示す．雑音は付加していない．a 軸の刻み数 α_{MAX}，b 軸の刻み数 β_{MAX}，r 軸の刻み数 γ_{MAX} いずれも共に 256 とした．これは a，b，r の検出精度を原画像 F 上の1画素単位（整数値）に設定したことによる．

CPU 時間（処理時間）は，通常方式では約 653

表 2 通常の計算方式による円の検出実験結果（$X=Y=256$，雑音なし）
Table 2 The result of the circles detection experiment by the usual method ($X=Y=256$, no noise).

黒点数 N	円検出結果（上段は円番号，下段は度数） 第1位	第2位	第3位	第4位	第5位	第6位	CPU 時間（秒）
851	#2 182	#3 180	#1 173	#5 164	#4 112	＊ 76	652.742

表 3 処理時間の比較（$X=Y=256$）
Table 3 The comparison of the CPU time ($X=Y=256$).

（本方式の処理時間）/（通常計算法の処理時間） N'	8	10	20	30	40	50	60	70	80	90	100
比率	1/340	1/452	1/424	1/230	1/108	1/53	1/31	1/18	1/12	1/7	1/5

表 4 必要メモリの比較（比率：通常の計算法とのメモリ比）($X=Y=256$)
Table 4 The comparison of the required memory ($X=Y=256$).

	本方式											通常の計算法（パラメータ空間使用）
N'	8	10	20	30	40	50	60	70	80	90	100	
必要メモリ量	960 Byte	2000 Byte	18.0 KB	63.7 KB	154.7 KB	306.6 KB	535.2 KB	855.9 KB	1284.4 KB	1836.3 KB	2527.3 KB	67MB
比率	1/73182	1/35127	1/3818	1/1078	1/444	1/224	1/128	1/80	1/53	1/37	1/27	

(秒)で，本方式の場合よりも時間がかかった．表3に処理時間の比較を示す．$N'=8$ の場合が $N'=10$ の場合よりも遅くなっているのはスーパコンのベクトル化率の差によるものと推定される．

表4に本方式と，通常の Hough 変換の計算方法における必要メモリ（原画像Fの格納用に必要なメモリを除く）の比較を示す．

通常方式ではパラメータ空間は前述のようにa軸の刻み数 α_{MAX}，b軸の刻み数 β_{MAX}，r軸の刻み数 γ_{MAX} いずれも共に 256 としたので，1語4バイトとして $4\times256^3\fallingdotseq67$ (MB) と非常に大きなメモリ量になる．

これに対し，本方式では必要メモリ量は式(11)のNを N' と読み替えて求めることができる．すなわち，サンプリング黒点数 N' に依存するが，最小の $N'=8$ の場合には約1KB弱で，最大の $N'=100$ の場合で約 2.5MB 弱となり，通常方式と比べると極めて少ないメモリ量となる．これらはパソコンでも工夫すれば十分実行可能なメモリの大きさである．通常方式と比べると $N'=8$ の場合で約 1/73000，$N'=100$ の場合で約 1/27 のメモリ量となる．

6. おわりに

大きなメモリ量を必要とするパラメータ空間を用いない方法（黒点ランダム抽出法）で，円を検出するための Hough 変換の計算法を提案し，基礎的な実験を行った．その結果，途切れや重なり，雑音のある大小5つの円をほぼ正しく検出することができた．また，パラメータ空間を使用する従来の通常の Hough 変換計算法との比較実験も行った．その結果，特に必要メモリ量には大きな差が現れた．すなわち，従来の方法では大型計算機でもメモリ確保が容易でない数十 MB という巨大なパラメータ空間を必要とするのに対し，本方式ではパソコンでも可能なわずか数 MB のメモリで計算を行うことができることが分かった．また，処理速度においては，サンプリング黒点数 N' をなるべく小さくすれば数倍～数百倍の高速化が可能なことが分かった．また，雑音にもかなり強いことが分かった．

本方式の特徴は従来提案されている Hough 変換をモディファイした様々な円抽出法に比べて，アルゴリズムが極めて単純で，Hough 変換の基本的な原理を踏襲したまま，省メモリと高速化を実現している点である．

今回はアルゴリズムのシミュレーション実験ゆえ，スーパコン等を用いたが，今後は，本方式を用いて原画像に実際の写真等を用いたパソコンベースの円検出実験を行いたいと考える．また，本方式を円だけでなく楕円や放物線，双曲線等の任意の2次曲線の検出，あるいはその他の数学図形の検出に拡張したいと考える．

謝辞 日頃研究活動に関し，種々御激励頂く大阪大学工学部通信工学科手塚慶一教授，大阪大学経済学部経営学科真田英彦教授に深謝する．また，本研究に関して，有益な御助言を賜った大阪大学産業科学研究所北橋忠宏教授に深謝する．

参考文献

1) Hough, P. V. C.: Method and Means for Recognizing Complex Patterns, U. S. Patent 3069654, December 18 (1962).
2) Duda, R. O. and Hart, P. E.: Use of the Hough Transformation to Detect Lines and Curves in Pictures, *Comm. ACM*, Vol. 15, No. 1, pp. 11-15 (1972).
3) 輿水大和：Hough 変換に関する最近の研究動向，情報処理学会研究会資料，CV 51-1, pp. 1-8 (1987).
4) 松山隆司，輿水大和：Hough 変換とパターンマッチング，情報処理，Vol. 30, No. 9, pp. 1035-1046 (1989).
5) Kimme, C., Ballard, D. and Sklansky, J.: Finding Circles by an Array of Accumulators, *Comm. ACM*, Vol. 18, No. 2, pp. 120-122 (1975).
6) Shapiro, S. D.: Properties of Transforms for Detection of Curves in Noisy Pictures, *Com-*

put. Gr. Image Process., Vol. 8, No. 2, pp. 219-236 (1978).
7) Tsuji, S. and Matsumoto, F.: Detection of Ellipses by a Modified Hough Transform, *IEEE Trans. Comput.*, Vol. 27, No. 8, pp. 777-781 (1978).
8) Sklansky, J.: On the Hough Techniques for Curve Detection, *IEEE Trans. Comput.*, Vol. 27, No. 10, pp. 923-926 (1978).
9) Conker, R. S.: A Dual Plane Variation of the Hough Transform for Detecting Non-concentric Circles of Different Radii, *Comput. Vision Gr. Image Process.*, No. 43, pp. 115-132 (1988).
10) 西江慎吾，伊東 晋，宇都宮敏男：一般化 Hough 変換の一高速化手法，テレビジョン学会誌，Vol. 43, No. 4, pp. 417-419 (1989).
11) 大橋靖弘，大和淳二，石井郁夫，牧野秀夫：一般化 Hough 変換による任意図形検出アルゴリズム，信学技報，PRU 88-123, pp. 33-39 (1988).
12) 輿水大和，村上和人：Hough 変換平面からの図形特徴抽出，1990 信学春季全大，D-521 (1990).
13) 輿水大和：直線パターン検出のための Hough 曲線追跡型アルゴリズムについて，信学論D，Vol. J 68-D, No. 10, pp. 1769-1776 (1985).
14) 恩田邦夫，青木由直：三角関数の周期性を利用した Hough 変換の高速計算法，信学論D，Vol. J 70-D, No. 10, pp. 2009-2011 (1987).
15) 安居院猛，崔 亨振，中嶋正之：ピラミッド階層を利用した高速ハフ変換について，信学技報，IE 86-67 (1986).
16) 花原啓至，丸山次人，内山 隆：実時間 Hough 変換プロセッサ，昭 60 信学情報・システム部門全大，No. 92 (1985).
17) 塩野 充：丸め併合法を用いた一般化 Hough 変換によるディジタル画像からの円図形の検出，1990 年情報学シンポジウム，pp. 27-38 (1990).

(平成2年 7 月 10 日受付)
(平成2年 11 月 13 日採録)

塩野 充（正会員）

昭和 22 年 (1947) 生．昭 47 大阪大学工学部通信工学科卒業．昭 56 同大学院博士後期課程修了．工学博士．昭 57 岡山理科大講師．昭 59 同助教授．昭 63 同教授，現在に至る．パターン認識，画像処理，CG の研究に従事．著書「新 JIS FORTRAN 入門」，「電子計算機基礎論第2版」，「コンピュータ & 通信」，「実践 N88 日本語 BASIC」，「ニューメディア概論」，「BASIC 画像処理プログラム 150 選」，「衛星放送とハイビジョン」ほか多数．電子情報通信学会，テレビジョン学会，人工知能学会，画像電子学会，神経回路学会，AVIRG 各会員．